Applying XML to Information Management

20 April 1999

Dale T. Anderson

dta@dtai.com

Outline

- Background
- Projects Looking for a Solution
- Organizational Involvement
- Information Management (IM) Technology
- LEIF
- SPAWAR N-Tier AppServer Project
- XML Schema Standards
- IM Requirements of XML
- Plan and Summary

Background

- DTAI has extensive experience in Information Management (IM), C2, and C4I development
- Recently completed LEIF 3.0, an Information Framework
- LEIF-like IM technology is being applied to Data Normalization for Horizontal Integration in the SPAWAR App Server effort
- Several programs have indicated a desire for an XML parallel to LEIF-like Information Representation

Projects Looking for a Solution

- DTAI already has several customers that desire XML information representation
- DARPA ISO Integrated Heterogeneous Applications project (Hank Oxford, Project Lead, hoxford@dtai.com)
- SPAWAR N-Tier App Server Project (Joe Mayo, Project Lead, mayoj@dtai.com)
- LEIF in general (ACOA, ALP, NGII, …, Jeff Herman, Project Lead, jsherman@dtai.com)

Organizational Involvement

- SPAWAR PMW-157 Common Data Store (CDS) Group
- DISA SSD-MD (Semi Structured Data and Meta Data) Subpanel
  - Subpanel of the DII COE Data Access Technical Working Group (TWG)
  - Launched December 1998
- Looking for input from Joint Message Handling System (JMHS)

SSD-MD Charter (implied)

- Develop specifications and/or DTDs
- Select metadata standards and tools
- Create DTD repository / distribution mechanism / versioning management
- Provide guidance for tag terminology
- Develop “enhanced” XML editors for coded XML docs
- Develop application interpreters for XML
- Reference implementations

IM Technology

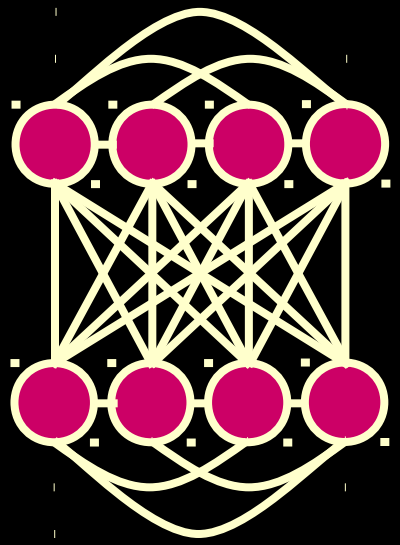
- DTAI's Information Management (IM) technology includes
  - Strategies for the semantic and structural representation of the information
  - Processes and paths of information flow
- A primary goal of our IM technology: To support the rapid development of *information integration solutions*

Targets of IM Technology

- Supporting better integration of DII COE products and data
- Horizontal Integration of SPAWAR Products
  - Dr. Frank Perry's brief on 10/8/98
  - Goal is *true integration*; not just having all of the apps available from the same screen
- Enterprise Application Integration solutions (new commercial buzzword associated with Application Server technology)

Information = Data + Semantics

To turn this…

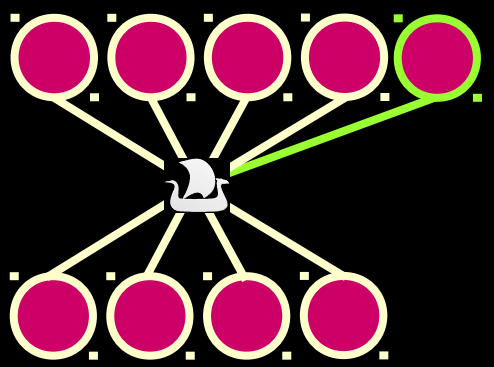


Supports the addition of a new module with only *one* new line of communication…

Increases capability…

Decreases complexity, effort, and *lifecycle cost* (development and maintenance)

… into this



Even with common protocols like CORBA, RMI, and COM, the mechanism is only half the problem. A common convention for semantic interpretation must still be applied.

Object integration does NOT EQUAL information integration

Blending Information

- Objects that can realize their common attributes can be *blended together* in limitless configurations (e.g., locations on maps)
- Each data source provides a “layer” of information
- Users will see new relationships and make better assessments
- Data fusion applications can more easily leverage patterns and relationships

A common semantic base leads to better integrati

Horizontal Integration of Systems

- Promoting *Information Sharing!*
- IM technology provides the common "vocabulary" by which heterogeneous applications and systems can communicate
- This is a *powerful enabler for Systems Integration* on a large scale
- Users can make new inferences based on the combined data set
- New applications can be developed to make better of a richer information base

Semantic Representation

- Object-oriented representation of "***data item***" objects
- Self-described data items
  - "Information Aware" components can leverage data sources never seen before
  - Typed attributes -- well-defined name/value pairs
  - Methods -- commands/services that the object can perform
- Domain Policy
  - Attributes, types, and methods known to exist
  - Supports domain-level interaction between components
  - A data item can comply with any number of Domain Policies at one time (e.g., "Geo Domain", "Temporal Domain", "C2 Domain")

Attributes and Methods

- Typed Attributes
  - New attribute types can be leveraged without requiring changes to existing software
  - Type Meta Data provides type name/description, text conversion, validity/ranges, numeric units, more…
  - Additional attributes (application defined…not domain)
    - Statically declared (pre-defined classes and interfaces)
    - Dynamically declared (determined and evaluated at runtime)
- Methods
  - User-oriented commands -- support dynamic generation of “command menus”
  - Domain Policy methods (i.e., declared method “interfaces”) can be leveraged by other software components

Semantic Representation (cont.)

- Relationships between data items
  - Direct references from one data item to another
  - Separately-defined associations and aggregates
  - Model any complex relationship/structure (Graph Theory)
- Data Integrity Support
  - Transactions
  - Locking
- Security Support
  - Authentication
  - Authorization

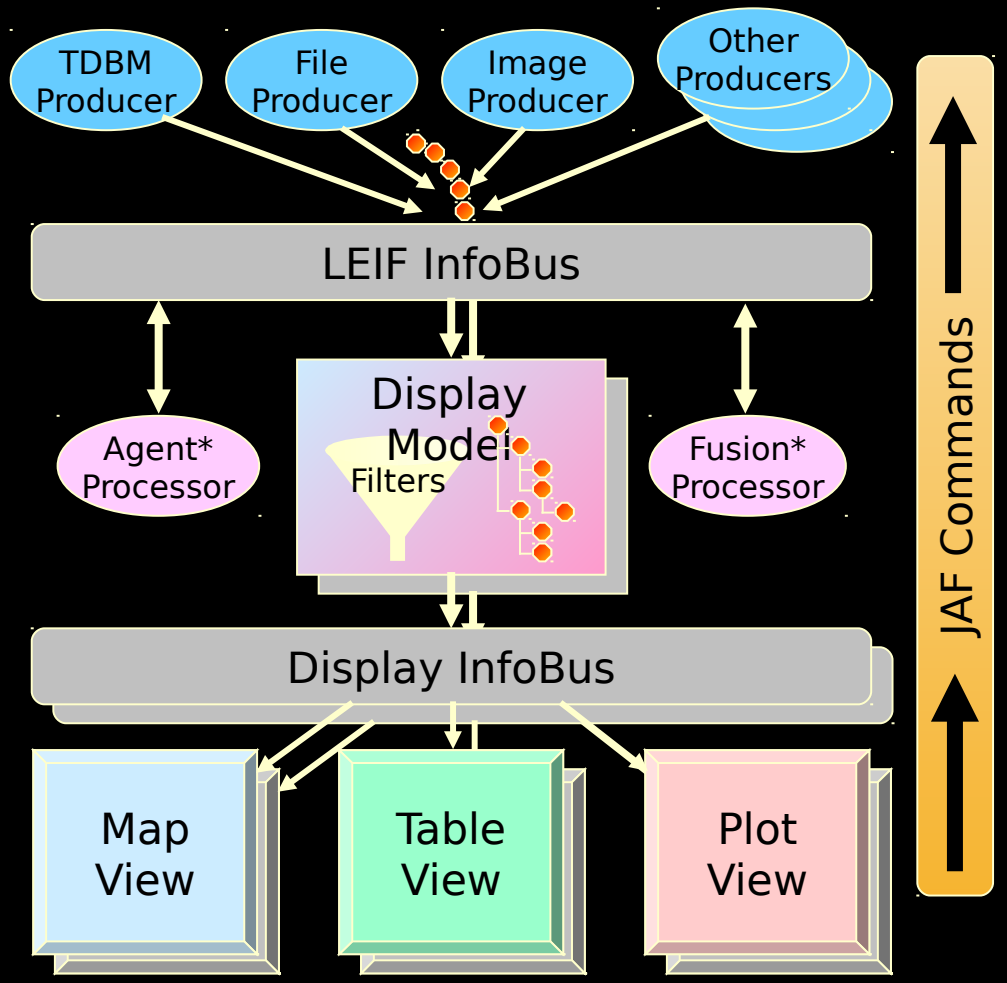
LEIF

- LEIF is a software framework for the development and deployment of modular plug and play applications
- LEIF supports rapid development of *information integration solutions*
- LEIF provides powerful data-independent visualization applications and the tools to develop new ones
- LEIF can be extended and molded into a wide range of vertical applications

Targeted Users

- LEIF is designed with two groups in mind
  - LEIF Operators (the end user): Make it easy to use but powerful in terms of visualization and data query, navigation, and selection
  - Software Developers: Make it easy to extend with new capabilities and easy to target to different domains
- Award Winning: On December 8, LEIF was awarded the *Java Developers Journal* Editors Choice Award in the Framework Category

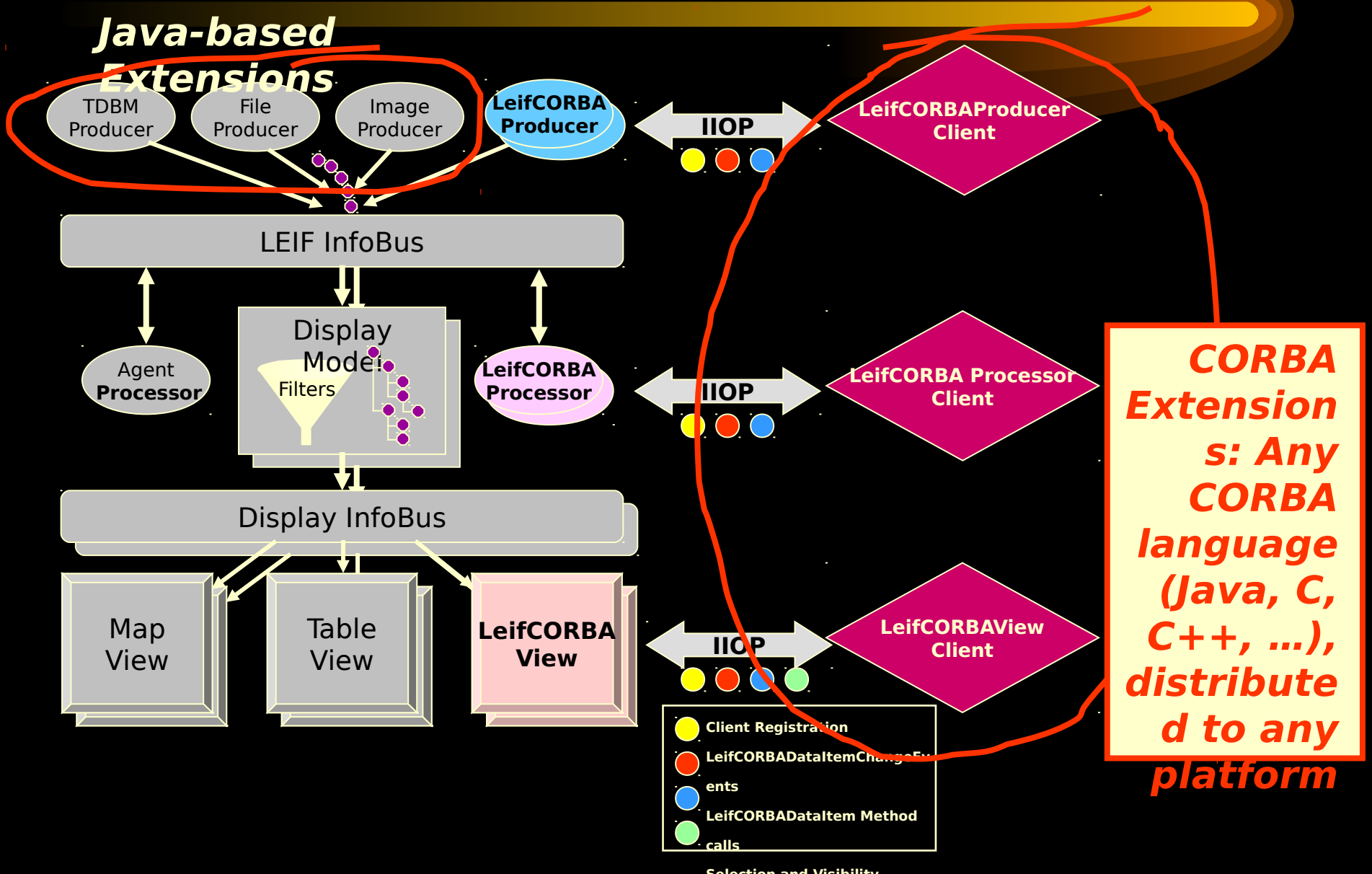
LEIF Architecture



*Agent and Fusion Processors are examples only. They are not real parts of LEIF at this time.

- LEIF is a Framework
  - Independently developed Extensions are "plugged in"
  - *Producers* interface to any data source, ANY PROTOCOL, and encapsulate data and semantics
  - *Processors* process produced data and enhance or augment the information
  - *Views* display any or all data in multiple configurations
  - Display Model - organizes and filters data for Views
- Leverages open industry standards (e.g., advanced JavaBeans technologies) and COTS APIs and products whenever possible

Java and CORBA Integration

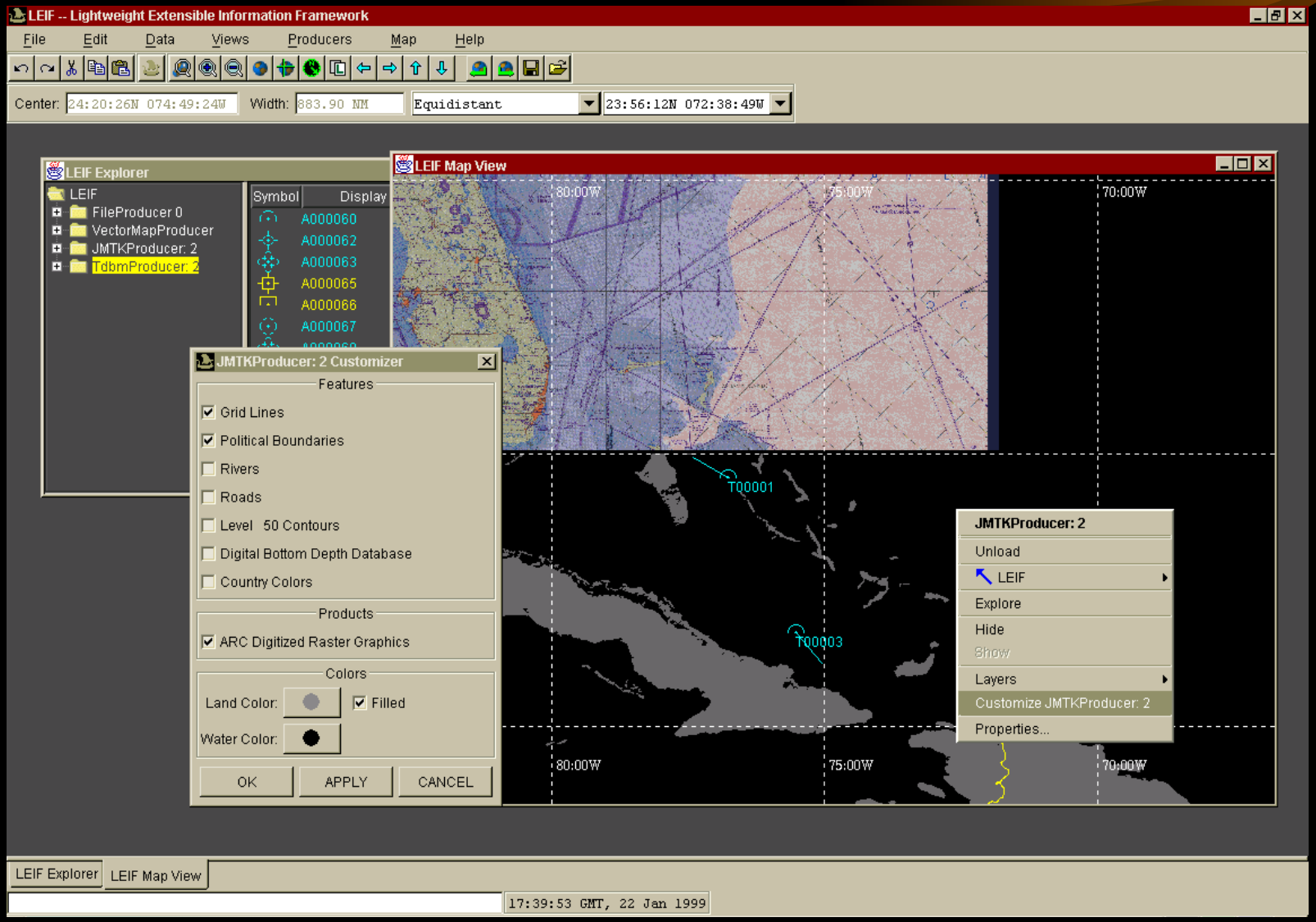


Core Extensions

- Producers
  - TDBM
  - Map (JMTK, Vector, ESRI IMS)
  - File
  - Registered Image
  - Database
- Views
  - 2D Map
  - Explorer (Tree/Table)
  - Plot (Scatter, Bar, Pie, Gantt)
  - Property

These are *some* of the extensions currently delivered with the basic LEIF distribution. Many other LEIF extensions have been developed by other Contractor and Government software

LEIF Producers and Views



LEIF and NGII (JTF-ATD)

- DARPA Technical Integration Experiment (TIE) underway to integrate LEIF and NGII
- NGII provides tools to read UML and ER diagrams to generate middle-tier servers
- Result of TIE should be NGII code generation of LEIF Producers
- Proposed extension to this: generate XML Schemas

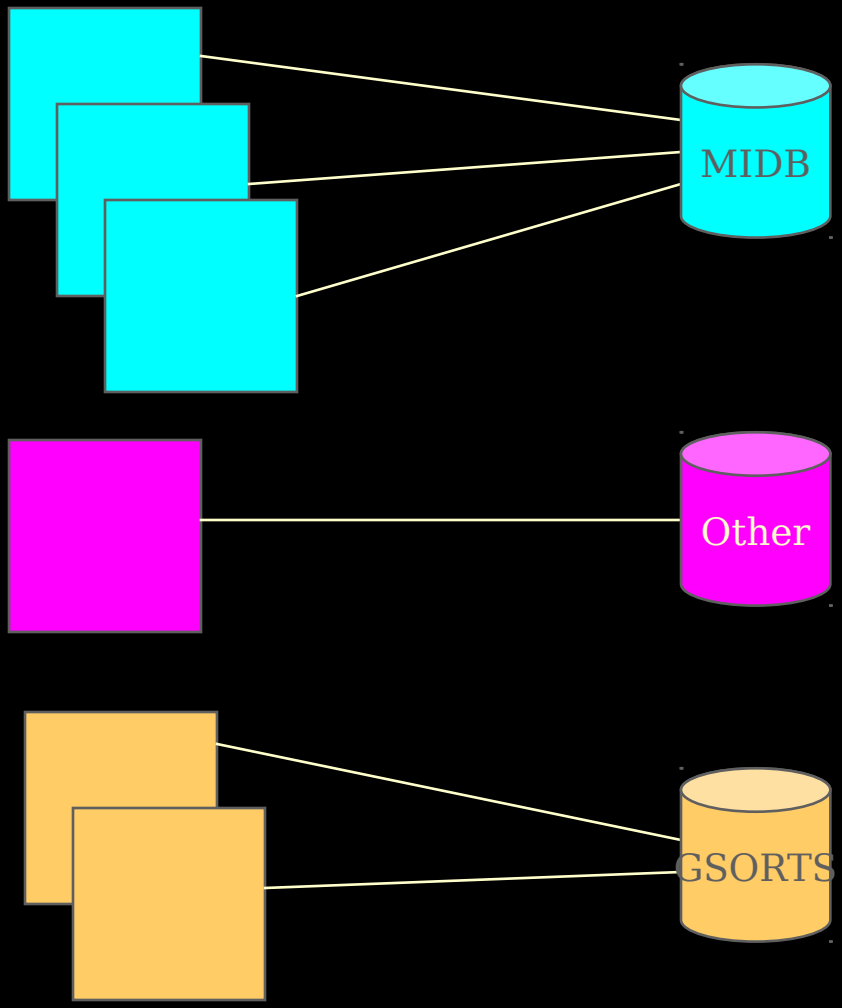
SPAWAR AppServer Project

- Design, Develop and Deploy a Secure N-tier Information Management and Dissemination Framework to allow the Horizontal Integration of Information.

Strategy

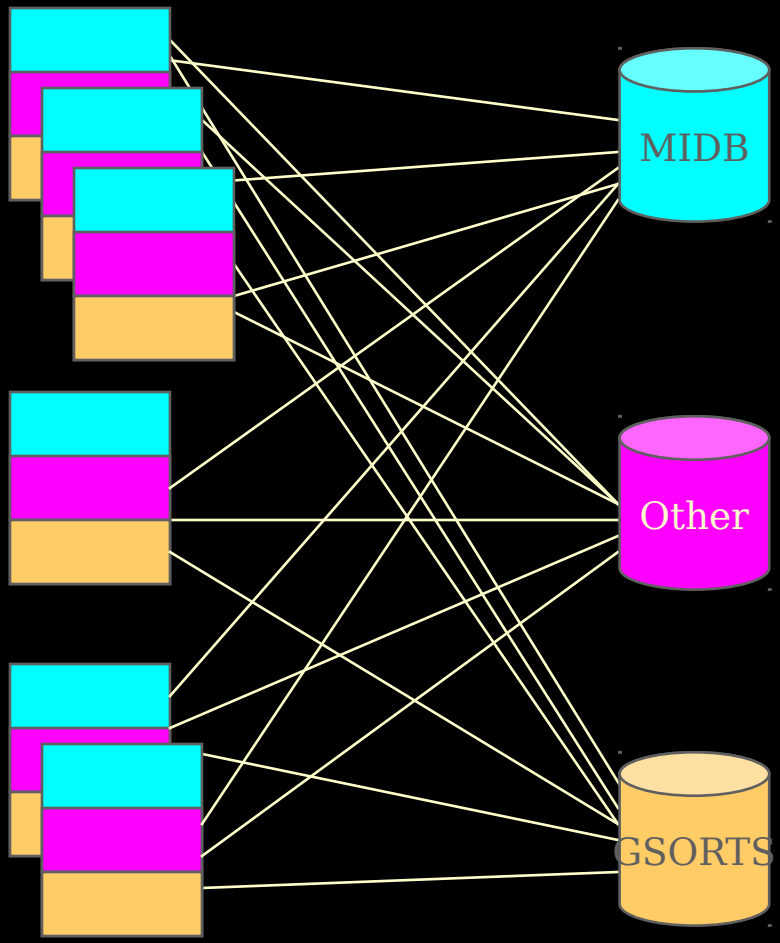
- Design in Security from the start
- Leverage COTS *Applications Servers,* use only their standards compliant features to avoid vendor lock-in.
- Separate Presentation, Business Logic, and Data Access of the Applications to extend over an N-tier Application Server Architecture.
- Implement Application Server *independent* “Horizontal Integration Data Layer” (Data Normalization and Event APIs) to create and obtain data in Self Describing Object or XML format.

Today's Client-Server Architecture



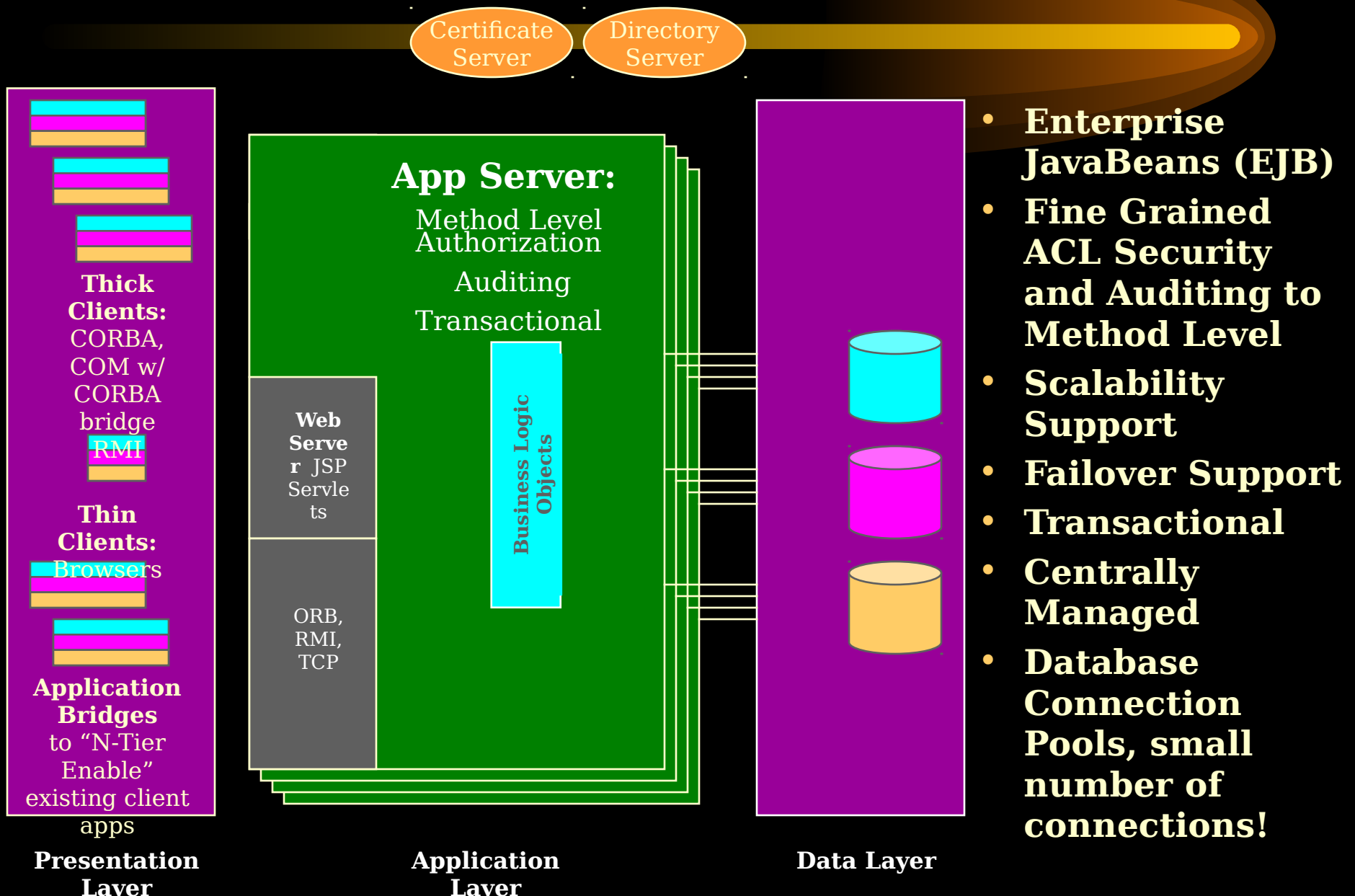
- Client Server Fat Apps... NO real scaling
- Stovepipe Systems... NO sharing of data

Current Architecture with Sharing



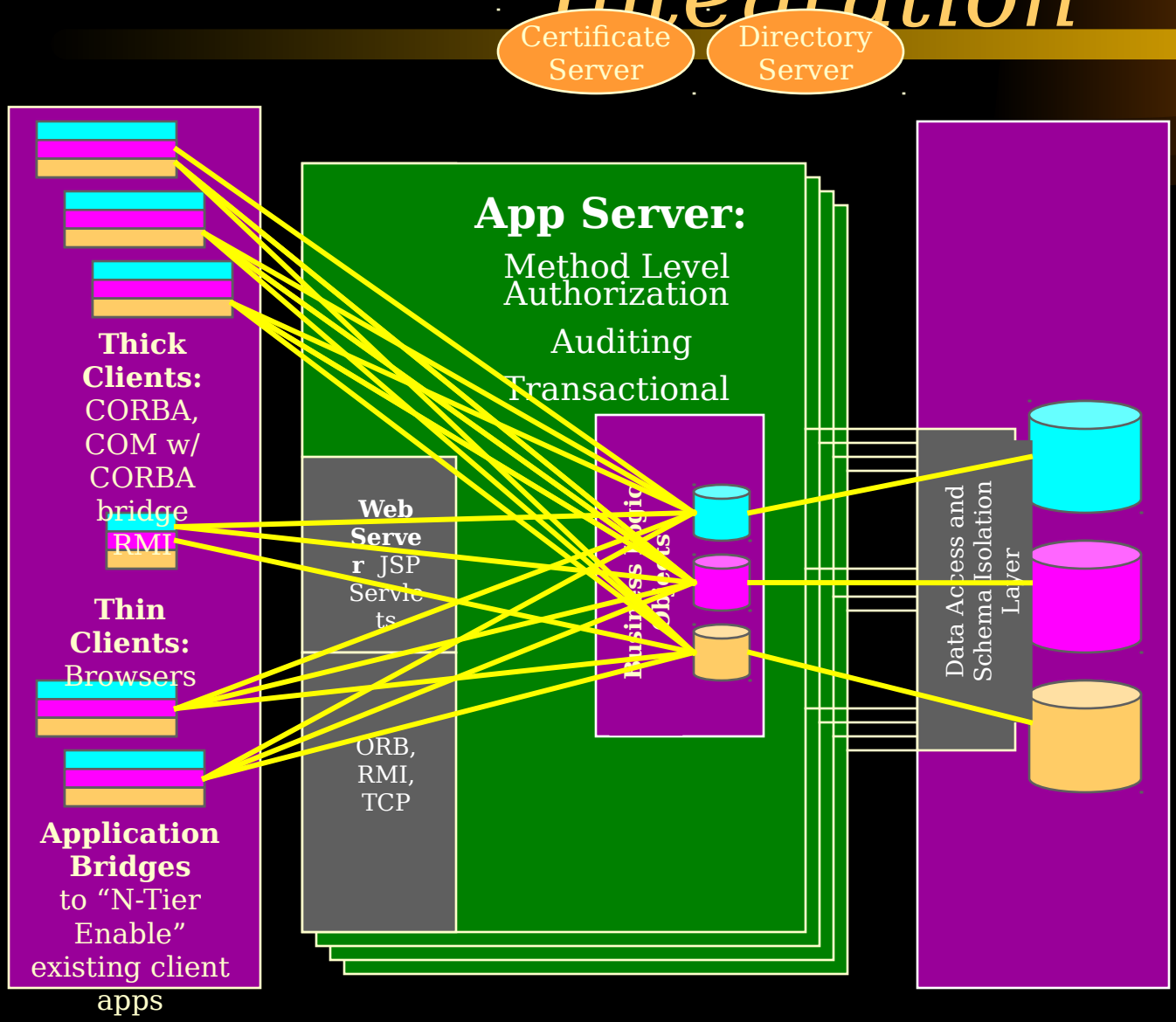
- In this environment, “sharing” means adding the semantics and business logic of a data source to every application interested in the data

App Server Architecture Goal



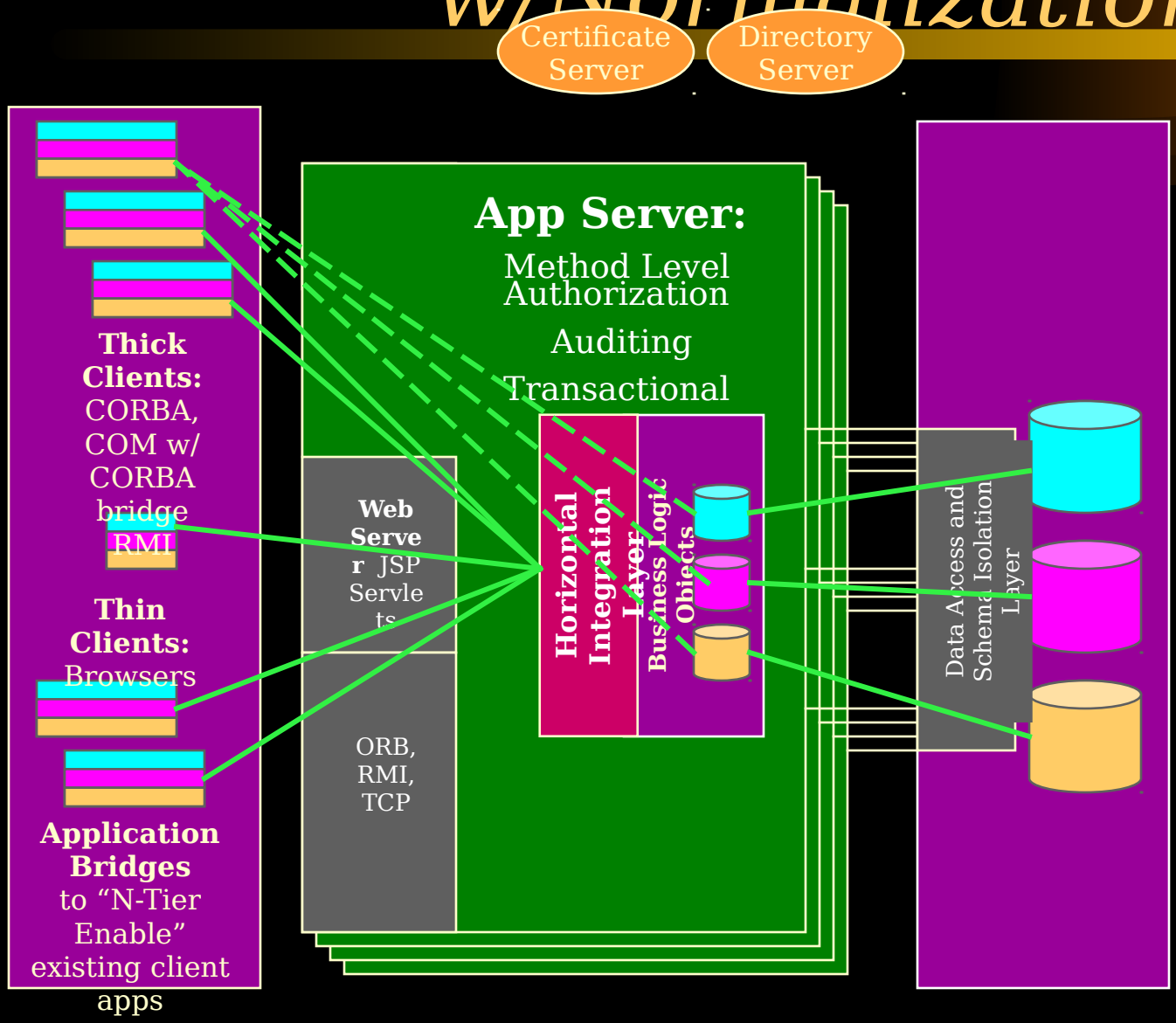
- **Enterprise JavaBeans (EJB)**
- **Fine Grained ACL Security and Auditing to Method Level**
- **Scalability Support**
- **Failover Support**
- **Transactional**
- **Centrally Managed**
- **Database Connection Pools, small number of connections!**

Requirement for Horizontal Integration



- **We still need to know specific API semantics of each Business object to invoke**
  - **Are these the same? speed, velocity, pace**
  - **Knots or KmPH?**
  - **Can I have a negative velocity?**
  - **How do I format/present this value?**
  - **What is this related to? (Drill-down, referenced data and URLs)**
  - **What can this or that object do for me?!**

N-tier Architecture w/Normalization



- **New data sources are easily added, and *immediately available* to any presentation level component enabled for normalized data**
- **Components for building client apps can be generalized based on "Information Awareness"**
- **Applications developed from these components can be targeted to specific domains, or built as general purpose information space navigation and analysis tools *as needed!***

AppServer Normalization Format

- The App Server should support multiple formats of the Normalized Data
  - Java object (also available via RMI)
  - CORBA object
  - XML “document” object
- App Server will supply a “translator” to convert normalized middle-tier-generated Java objects into any of these other formats

Schema vs. DTD

- XML Document Type Definition (DTD)
  - Current standard for defining XML grammars
  - Scope is limited to basic document structure (tag names, attributes, sequence, etc.)
  - No validation or implied interpretation of content!
- XML Schema
  - Extends DTD to define documents as well-described data elements
  - DTD's declare elements…Schemas *define* them
  - Makes *content* validation possible

XML Schema Standards

- W3C Recommendation is many months away
  - Requirements for an “XML Schema” standard were just released in February
  - Several competing schema-oriented “Notes” have been submitted to the W3C
- Three leading Notes:
  - DCD (Document Content Description)
  - DDML (Document Definition Markup Language)
  - SOX (Schema for Object-oriented XML)
- Little to no formal support (i.e., software) for these formats (more tools to come to actually generate useful output)

Summary of existing languages

Language	Pros	Cons
DDML	Simple Complete spec	No data types Limited reuse
DCD	Elegant reuse model Simple except...	Spec not quite complete Must learn RDF
SOX	Feature rich Namespaces not standard Many small holes in spec	Complex
XML-Data In MS parser	Feature rich	Complex Many holes in spec
W3C	Eventual winner	Not available

Purpose of a "Schema"

- From "XML Schema Requirements" 15 Feb 99 (http://www.w3c.org/TR/NOTE-xml-schema-req)
  - To define and describe a class of XML documents … to constrain and document the meaning, usage and relationships of … datatypes, elements and their content, attributes and their values, entities and their contents and notations.
  - Schemas document their own meaning, usage, and function. [self-describing]
  - Used to define, describe and catalogue XML vocabularies…
  - To express syntactic, structural and value constraints…

Applicability to Current Needs

- SOX and DCD appear to address many of the syntactic needs of our IM concepts
  - Still preliminary…need more time to evaluate
  - Risk in adopting one of these “Notes” when the standard has still not been formalized
- Format extensibility is mandatory (especially if not all needs are met by the basic standard)
- We do NOT want to “re-invent”

IM Requirements of XML

- Derive a DTD from the Schema Definition
  - Existing Schema formats may (likely) have their own Schema-to-DTD translation
  - Risk is that the implied DTDs may not meet our goals (e.g., concise)
- Schema and generated DTDs must support all existing IM capabilities

Design Objectives

- Small as possible XML data “documents”
  - Storage, bandwidth, time to parse
  - Pay only for the features you use
- Fully describe information semantics
  - Data, relationships, and commands
  - Use LEIF as a reference implementation
- Two-way communication (not just an input format)
  - Read *and Write back* objects (update, create, delete objects)
  - Get *and Set* attributes
  - Execute methods on the data items (defined by type *or* instances)
- Extendable *and Evolvable* Schema Format
  - Pie in the sky? Perhaps…
  - There are always new ideas…*plan for change!*

Required IM Features

- Objects, interfaces, and inheritance
- Attributes (name/value pairs)
  - Domain (well-known) based on “domain policy” (and nested domains)
  - Self-described additional attributes
- Type Meta Data: Completely define the attribute type
  - From String and number to date/time, color, icon, and new types
  - Semantics and constraints (e.g., type name, units, ranges)
  - Rendering and editing support
    - At least basic formatting and regular expressions
    - As much as possible without code? (Or do we figure out how to incorporate code…Java or ECMAScript, XFDL, DHTML/DOM?)
- Relationships between objects (XML or non-XML objects)
  - Attributes that represent relationships (URL-like form)
  - Data items that encapsulate relationships (associations, aggregates)
- Client-side object manipulation and method invocation

Influencing the Standards

- Results of initial research and requirements should be fed back into W3C XML Schema committee
  - Ideally, well before the XML Schema standard is ratified
  - Potential to influence the standard so that standard can be used unmodified
- Must be a W3C “member” to participate
  - DARPA supports (has supported?) the W3C
  - DISA is a current W3C “member”

Conceptual Example

(NOT A Proposal!)

Schema (some industry standard):

```
<CLASS name="Track">
    <IMPLEMENTS name="GeoDomain"/>
    <IMPLEMENTS name="TemporalDomain"/>
    <ATTRIBUTE id="Velocity" range="cont"
        units="Knots" defaultVal="0.0"
        type="float"/>
    <COMMAND id="Delete"/>
</CLASS>
```

Instance of this schema definition:

```
<DODXML>
    <Track key="1234124">
        <Velocity v="24.5"/>
        <LatLonAlt v="75 25 59N; 142 18 59W"/>
        <Time v="18:02:15"/>
        <SomeArray>
            39.9, 7222.1
        </SomeArray>
    </Track>
</DODXML>
```

Plan

- Select a candidate XML Schema format
  - Extend if needed to meet IM objectives
  - Develop a prototype schema and example data
- Demonstrate capability to SSD-MD and CDS Groups
  - Elicit feedback
  - Looking for an endorsement
  - Hopefully this will benefit all common efforts
- Integrate with LEIF, App Server, and other efforts as current “standard” XML IM format

Summary

- Our current IM technologies have demonstrated flexibility and power in defining extensible information systems and information flow
- XML is widely recognized as an excellent common format for information
- The combination of common semantic representation of information with a common medium for transmission and storage should yield long-term rewards in the IM domain

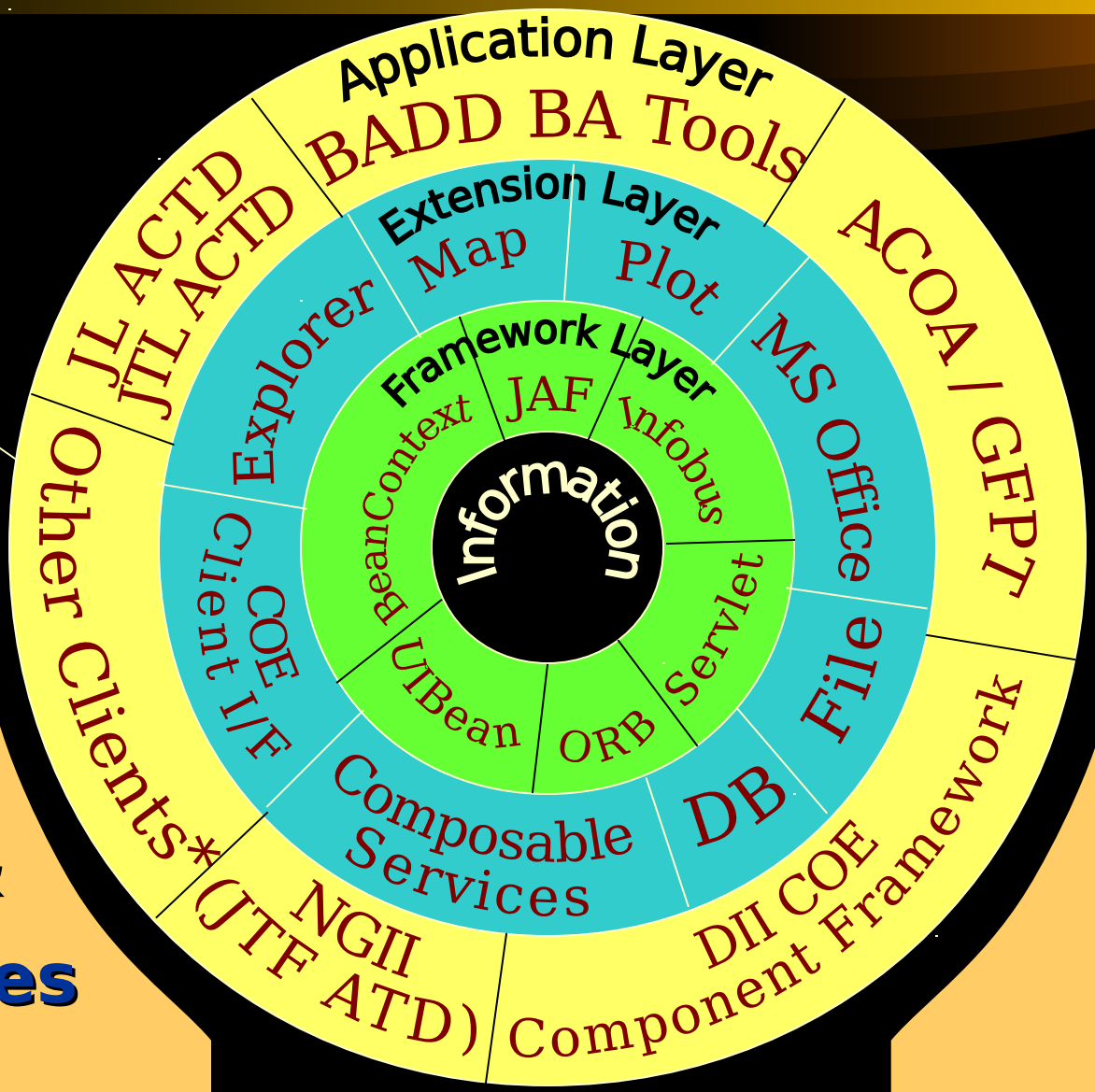
Backup Slides

Some Detailed Requirements

- Domain Policies must be declared in namespaces to deconflict similar policies (use Java-like “packages”, e.g., mil.disa.domains.c2domain)
- What Dictionaries or Repositories?
  - Base types (definition, libraries--jars and dlls)
  - Domain Policies
  - Units of measure and conversions
  - Versioned Schemas

LEIF-Enabled Programs

IW
DIA MOBA
CDS WEB
AADC
C2WCM
TADMUS
FP 2000
CPOF



COE APIs & Services

Other Databases & Sources